Exposing MPI Objects for Debugging

Brock-Nannestad, Laust; DelSignore, John; Squyres, Jeffrey M.; Karlsson, Sven ; Mohror, Kathryn

Publication date:
2014

Document Version
Publisher's PDF, also known as Version of record

Exposing MPI Objects for Debugging

Laust Brock-Nannestad*, John DelSignore†, Jeffrey M. Squyres‡, Sven Karlsson*, Kathryn Mohror§
*Technical University of Denmark
Email: {laub|svea}@dtu.dk
†Rogue Wave Software, Inc.
Email: John.DelSignore@roguewave.com
‡Cisco Systems, Inc.
Email: jsquyres@cisco.com
§Lawrence Livermore National Laboratory
Email: kathryn@llnl.gov

***Abstract*—Developers rely on debuggers to inspect application state. In applications that use MPI, the Message Passing Interface, the MPI runtime contains an important part of this state. The MPI Tools Working Group has proposed an interface for *MPI Handle Introspection*. It allows debuggers and MPI implementations to cooperate in extracting information from MPI objects. Information that can then be presented to the developer. MPI Handle Introspection provides a more general interface than previous work, such as Message Queue Dumping.**

We add support for introspection to the TotalView debugger and a development version of Open MPI. We explain the interactions between the debugger and MPI library and demonstrate how MPI Handle Introspection raises the abstraction level to simplify debugging of MPI related programming errors.

I. Introduction

Developers rely on debuggers to determine if their applications are behaving correctly. The debugger provides the link between the original source code and the executing code. A runtime such as MPI adds a new layer of information that is not inspectable by the developer, but still of interest in debugging. A developer cannot easily inspect the state of the MPI runtime with a conventional debugger.

Developers with access to an MPI implementation's source code can debug the runtime in a traditional fashion: by stepping through the code. This is time consuming and beyond of the scope of many developers. It also requires an understanding of the data structures used by a given MPI implementation.

In most cases developers have no interest in this level of detail. Instead, they want the MPI *API-level state*. This is the state of the application at the level of the MPI specification, and not the implementation. Debugging is simplified when this abstract MPI state is presented to the developer. With MPI handle introspection MPI object data is presented in an implementation agnostic way, simplifying the debugging process.

Using the proposal from the MPI Tools Working Group [1], we successfully add support for introspection to a development version of the TotalView debugger and to Open MPI. We implement both the debugger and MPI sides of the interface.

The rest of this abstract and our poster describes the interface and the benefits of simplified MPI debugging. We also show queries on MPI objects performed through our implementation.

II. Debugger support for introspection

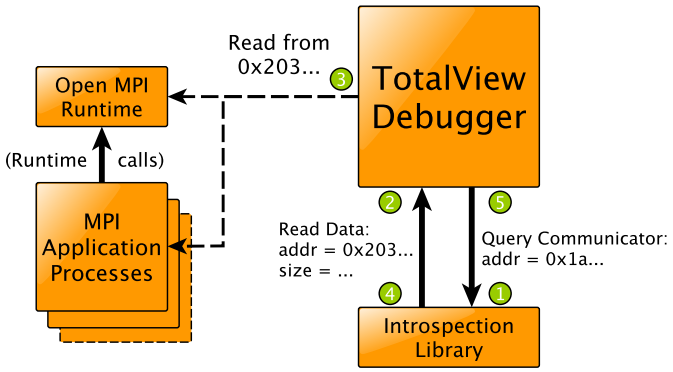

Fig. 1. Interactions between the debugger, MPI library, and application

The introspection interface builds on top of the well-established *Message Queue Dumping*, or MQD interface [2], which has been widely supported by MPI implementations for over a decade. The MQD interface was designed to simplify the problem of many different MPI implementations interfacing with many different debuggers. It provides a standardized low level interface between the two.

The interface is based on *callback functions* provided by the MPI implementation and the debugger: It defines a set of functions, that the debugger must provide, to allow access to the application being debugged. In turn, the MPI implementation provides callbacks which allow the debugger to perform queries on MPI objects and data types. The MPI implementation is responsible for parsing its own data structures, hence it performs *introspection*.

The overall design is outlined in figure 1. The figure shows a simple use case where a debugger is querying the state of an MPI communicator. The execution flow passes back and forth between the debugger and the introspection library, as indicated on the figure:

1) The debugger queries a communicator, providing a handle (an address) identifying the communicator.

2) In order to provide this information, the library requests raw data from inside the application process. This will be the underlying data structure for the communicator and anything else that is required.
3) The debugger extracts the raw data from the process and returns it to the library.
4) The library parses the data and returns communicator information in a standardized format.
5) The debugger presents the information to the developer.

III. Benefits of Introspection

With introspection, the developer can achieve a deeper understanding of the application's MPI behavior without resorting to debugging of the MPI implementation itself. A couple of use cases are:

- The ability to link processes to ranks in different communicators helps the developer decide where to focus debugging efforts.
- Knowing the size and location of send and receive buffers can simplify debugging of memory problems such as buffer overruns.

From the perspective of the debugger vendor, a standardized interface simplifies support for many different MPI implementations and a richer experience for the user. For the MPI vendor, it can expose the MPI state without providing source code to the MPI library.

IV. Integration into TotalView

Our implementation is integrated into a development version of TotalView. TotalView interacts with a reference implementation of the introspection library that we implemented as part of Open MPI. A screen shot showing initial setup and loading can be found in figure 2. The aim is to support TotalView's graphical user interface, but currently support is integrated into the command line debugger. A transcript of a debugging session can be seen in figure 3. With the aid of the introspection library, information such as rank, name, type, local, and remote processes can be presented to the developer.

V. Conclusions and Future Work

We present examples where exposing the internal state of MPI objects is useful for debugging and we successfully implement the interface for *MPI Handle Introspection*. Our implementation allows the TotalView debugger to extract information concerning MPI communicators from applications that use Open MPI. This is more efficient than extracting the same information manually. Our implementation is integrated into TotalView's command line environment, but the aim is to fully integrate it into the graphical user interface. This remains as future work.

Acknowledgment

The authors would like to thank Adam Moody at Lawrence Livermore National Laboratory for useful insights on MPI debugging. This work (LLNL-POST-658417) was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344, and co-funded by the European Artemis PaPP Project nr. 295440 and COPCAMS project nr. 332913.

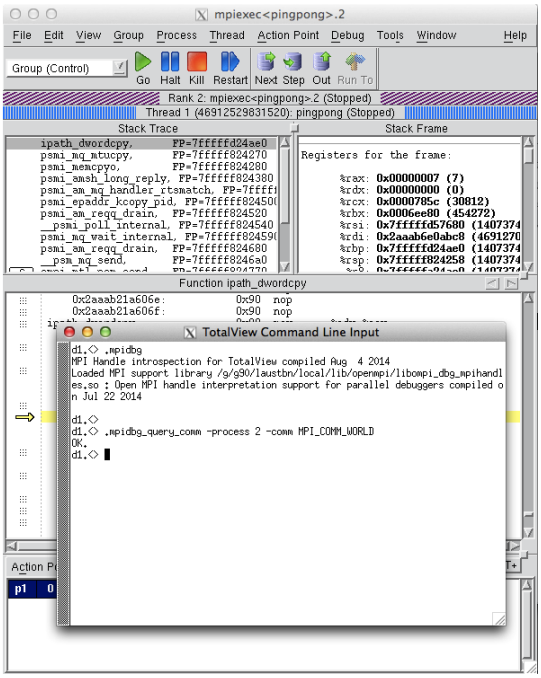

Fig. 2. Loading the introspection library into TotalView and requesting information on a communicator.

```
d1.<> .mpidbg
Loaded MPI support library /g/g90/laustbn/local/lib/
openmpi/libompi_dbg_mpihandles.so :
Open MPI handle interpretation support for parallel
debuggers compiled on Sep 5 2014

Finished loading MPI introspection support.

d1.<> dfocus p2
p2.<
p2.<> .mpidbgdump
Name                          Handle
MPI_COMM_WORLD                0x6028a0
MPI_COMM_SELF                 0x2aaaab01aa00
MPI_COMM_PARENT               0x2aaaab01a9e0
MPI_COMM_NULL                 0x2aaaab01a3e0

p2.<> .mpidbgquery basic 0x6028a0
Querying communicator 0x6028a0 in process 0x4878780
Communicator: MPI_COMM_WORLD
Rank: 0
Size: 4
Flag                                    Value
MPIDBG_COMM_INFO_PREDEFINED             True
MPIDBG_COMM_INFO_CARTESIAN              False
MPIDBG_COMM_INFO_GRAPH                  False
MPIDBG_COMM_INFO_TOPO_REORDERED         False
MPIDBG_COMM_INFO_INTERCOMM              False
{...}
Query was successful
```

Fig. 3. Transcript of a debug seesion with queries on an MPI Communicator.

References

[1] "MPI Tools Working Group," https://svn.mpi-forum.org/trac/mpi-forum-web/wiki/MPI3Tools, October 2014.
[2] J. Cownie and W. Gropp, "A Standard Interface for Debugger Access to Message Queue Information in MPI," in *Recent Advances in Parallel Virtual Machine and Message Passing Interface*. Springer, 1999, pp. 51–58.